Sistemas de arquivos

Objetivos:

1. Persistência: permitir o armazenamento de grandes quantidades de informação;
2. Acesso: permitir o acesso seguro de vários aplicativos e usuários;
3. Organização.

A informação será armazenada em uma unidade abstrata chamada *arquivo*. Os arquivos normalmente são agrupados em diretórios (ou *pastas*)dentro de uma estrutura hierárquica organizada de acordo com a similaridade de conteúdos. A Figura 1 mostra a organização básica de diretórios do sistema operacional Unix.

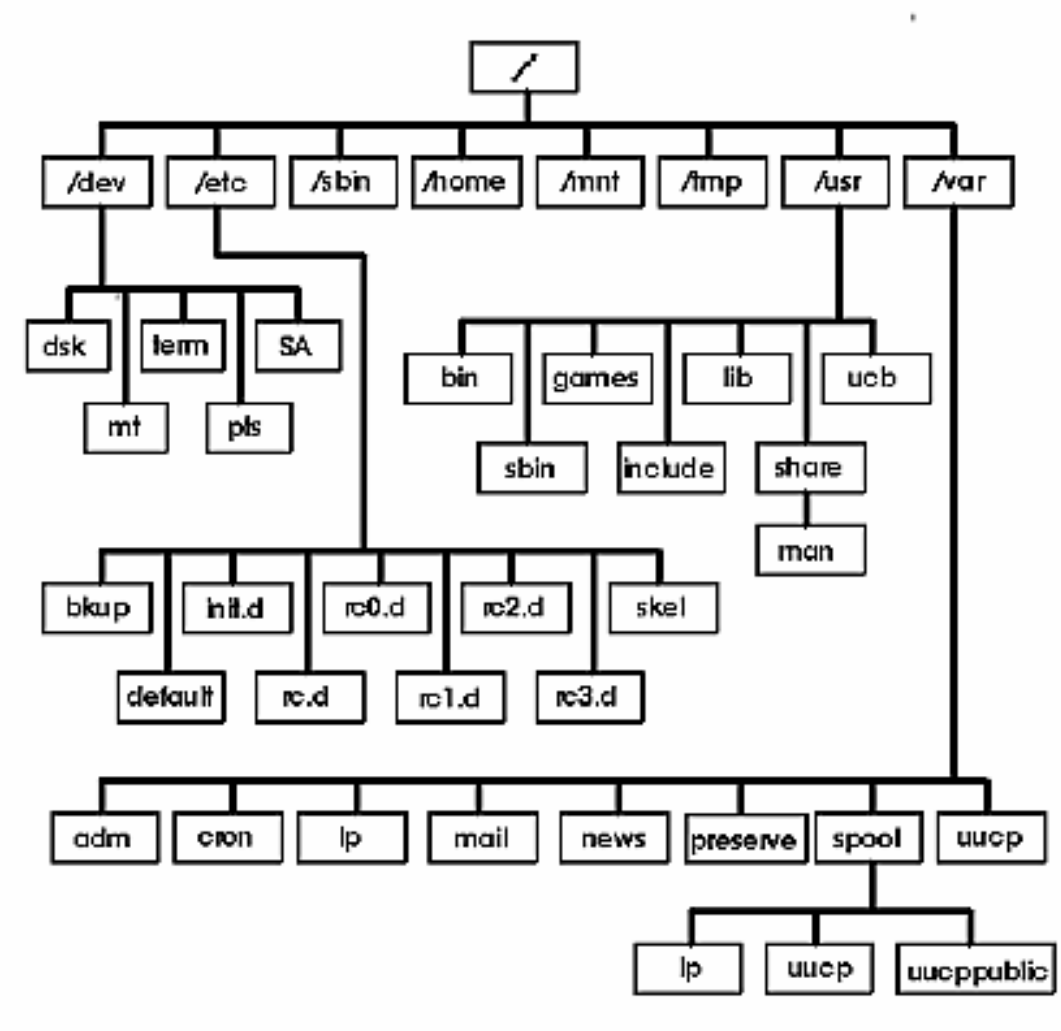


Figura : Estrutura de diretório do Unix System V

Operações sobre arquivos

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Seek: desloca o ponteiro de leitura/escrita para um determinado ponto do arquivo;
8. Set/Get atributo: lê ou modifica um atributo do arquivo.

O disco rígido

O disco rígido ideal possui:

1. Grande capacidade de armazenamento;
2. Rapidez no acesso a informação;
3. Baixo custo;

É muito difícil atingir tais objetivos, uma vez que a definição de *grande capacidade*, *rapidez* e *baixo custo* também varia conforme a tecnologia progride.

Os seguintes conceitos são fundamentais para compreensão do funcionamento do disco rígido:

- Trilhas
- Setores
- Discos e faces
- Cilindro
- Tempo de seek
- Latência de rotação
- Taxa de transferência

Figura : Disco rígido aberto

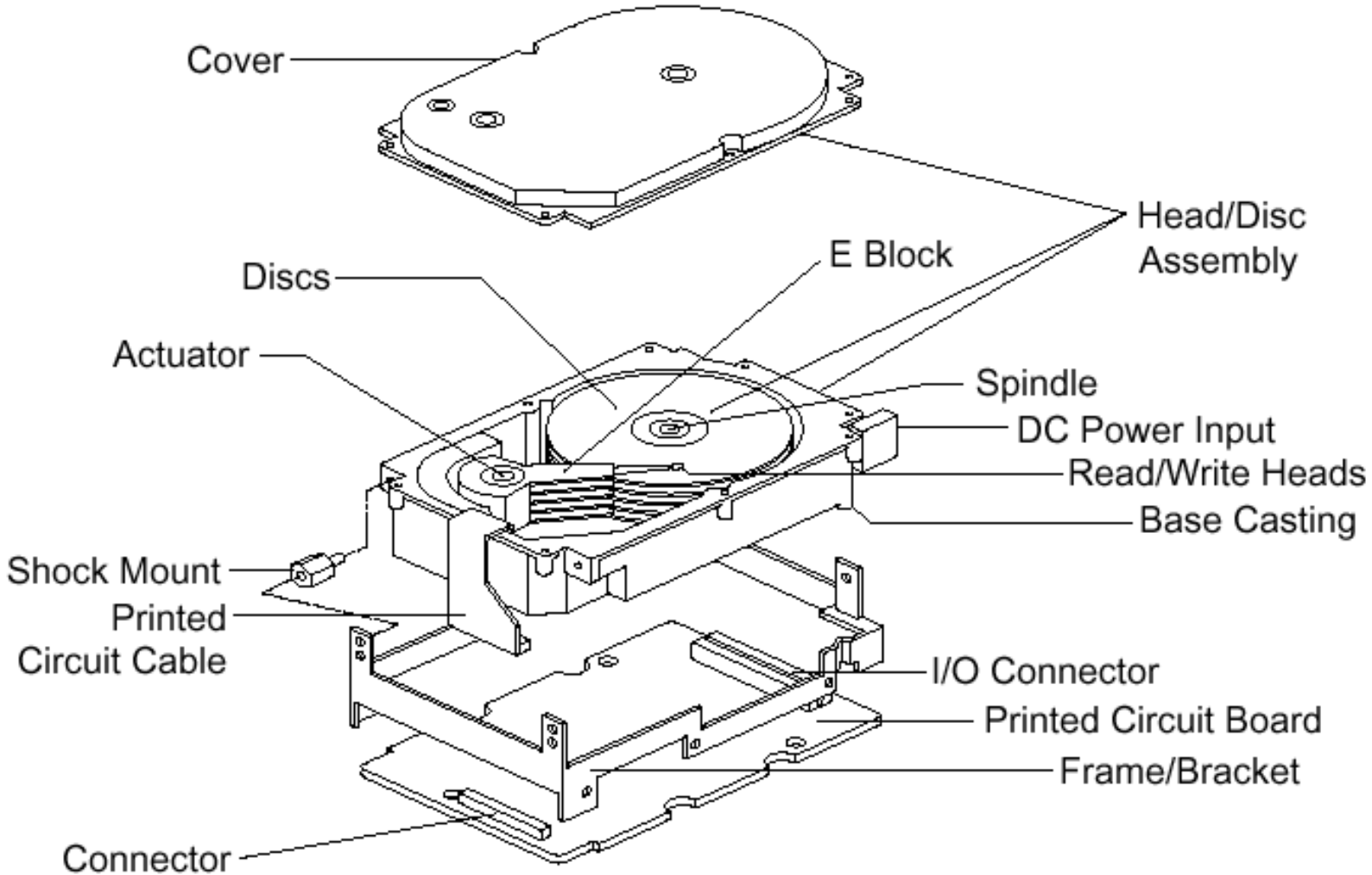
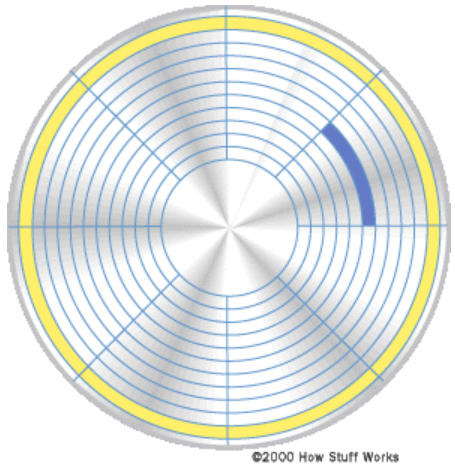


Figura : Elementos do disco rígido

Disco rígido: conceitos

Conforme a tecnologia avança, o espaço gasto para armazenar um bit é decrementado: a densidade de informação aumenta. Isso faz com que os cilindros estejam mais próximos.

Discos modernos possuem mais setores nos cilindros externos do que os cilindros internos. Os discos modernos possuem eletrônica e software que escondem a complexidade da implementação mecânica e dão a ilusão do mesmo número de setores em todas as trilhas



Tamanho do bloco

O fabricante do disco rígido escolheu o tamanho do bloco físico; normalmente não há como alterar este valor. Tipicamente, o tamanho do bloco físico é de 512 bytes. No entanto, o sistema operacional pode decidir utilizar como a menor unidade de leitura outro tamanho bloco lógico ou cluster, com tamanho múltiplo do bloco físico.
Utilizar um bloco lógico grande melhora o desempenho do sistema, pois isso tende a diminuir a fragmentação dos arquivos. No entanto, um bloco lógico grande pode levar ao desperdício de espaço (um arquivo pequeno ocupa no mínimo um bloco).
Alguns valores típicos:

- tempo de seek: 10ms
- taxa de transferência para blocos contíguos: 10K bytes por ms.

A FAT (*File Allocation Table*) é utilizada pela Microsoft nos sistemas operacionais Windows, apesar da disponibilidade do NTFS (*NT File System*. A FAT também é chamada de TAA (*Tabela de Alocação de Arquivos*.
Quando a partição do disco é formatada com a FAT, o sistema divide o disco em 4 regiões: MBR (*master boot record* ou registro de boot), FAT, diretório raiz e blocos de dados.

Registro de boot O primeiro setor do disco armazena o registro de boot. Este registro contém informações sobre o disco propriamente dito e suas partições. Através das informações contidas no registro de boot, os dados armazenados poderão ser acessados. Além disso, no registro de boot também existe um pequeno programa responsável pela inicialização dos sistema operacional, que será acionado pela própria BIOS do computador no instante da inicialização do sistema. São informações contidas no registro de *boot*: Bytes por setor. Setores por cluster. Número de setores reservados. Número máximo de entradas do diretório raiz. Número total de setores. Descrição da mídia (é um disco rígido?). Número de setores alocados paraFAT. Número de cabeças de leitura. Número de setores escondidos. Programa de *boot*.

FAT

A FAT é na verdade uma lista representado o encadeamento de blocos utilizada por um determinado arquivo armazenado no disco. Desta maneira, o sistema operacional pode armazenar os dados do arquivo em qualquer bloco do disco e recuperar facilmente o encadeamento do arquivo.
Na região de diretório raiz é armazenada a entrada inicial da FAT utilizada pelo arquivo. São alguns valores possíveis a serem empregados na FAT-16:

- Cluster livre: 0000;
- Cluster defeituoso: FFF7;
- Último cluster do arquivo: FFFF.

Existem 3 versões para a implementação da FAT: FAT-12, FAT-16 e FAT-32, referindo-se ao número de bits utilizada em cada entrada da FAT. A FAT-12 foi utilizada em disquetes e a FAT-32 é utilizada em sistemas atuais. A FAT-16 não é utilizada em função da limitação que impõe ao tamanho do disco (suporta discos com até $2^16 = 65536$ clusters. Por segurança, existem duas cópias da FAT.

Diretório Raiz

O Diretório raiz armazena as informações sobre o diretório raiz. A informação é dividida em registros com o formato:

Nome Nome do arquivo com 8 bytes

Ext Extensão do arquivo, com 3 bytes

Atributo Atributo do arquivo, pode ser: Arquivo, Read-Only, System, Hidden, Volume, Directory.
Também são utilizados 1 byte reservado para uso com NT ou Novell, data de criação, data de modificação, data de acesso

Primeiro cluster Indica qual o primeiro cluster ocupado pelo arquivo. Esta referência será seguida na FAT para determinar-se o encadeamento de arquivo no disco. Ocupa 2 bytes;

Tamanho do arquivo Indica o tamanho do arquivo.

O sistema FAT-32 precisou modificar a estrutura de diretório. Basicamente, nomes longos são suportados escrevendo-se um atributo de arquivo incompatível (ou inválido) para o sistema antigo, trocando-se então a estrutura da entrada de diretório para armazenar o nome longo.
Os sub-diretórios são implementados marcando-se um arquivo comum como diretório e armazenando-se uma estrutura de diretório.

Tamanho *default* para formatação de partições de disco operando com a FAT-32:

Tamanho da partição	Tamanho do cluster
32 MB a 64 MB	512 bytes
64 MB a 128 MB	1 KB
128 MB a256 MB	2 KB
256 MB a 8GB	4 KB
8GB a 16GB	8 KB
16GB a 32GB	16 KB
32GB a 2TB	Not supported

Tamanho *default* para formatação de partições de disco operando com a exFAT:

Tamanho da partição	Tamanho do cluster
7 MB a 256 MB	4 KB
256 MB a 32 GB	32 KB
32 GB a 256 TB	128 KB

Fonte https://support.microsoft.com/en-us/kb/140365

Nós I (ou I-Nodes)

Este sistema baseado em nós índice (*I-Nodes*) é utilizado em sistemas Unix/Linux. Quando o sistema é formatado, a partição é dividida em 4 partes:

Registro de boot descrito anteriormente. No entanto, o sistema Unix não utiliza as informações armazenadas no registro de boot. O registro de boot, neste caso, é utilizado apenas para inicializar o sistema.

Superbloco contém informações sobre a quantidade de nós índice, número de nós índice livres e ocupados, tamanho do cluster, tempo desde a última montagem, última checagem do sistema de arquivos, intervalo máximo entre checagens, mapa de bits para indicar os blocos livres no sistema.

Nós I (ou I-Nodes)

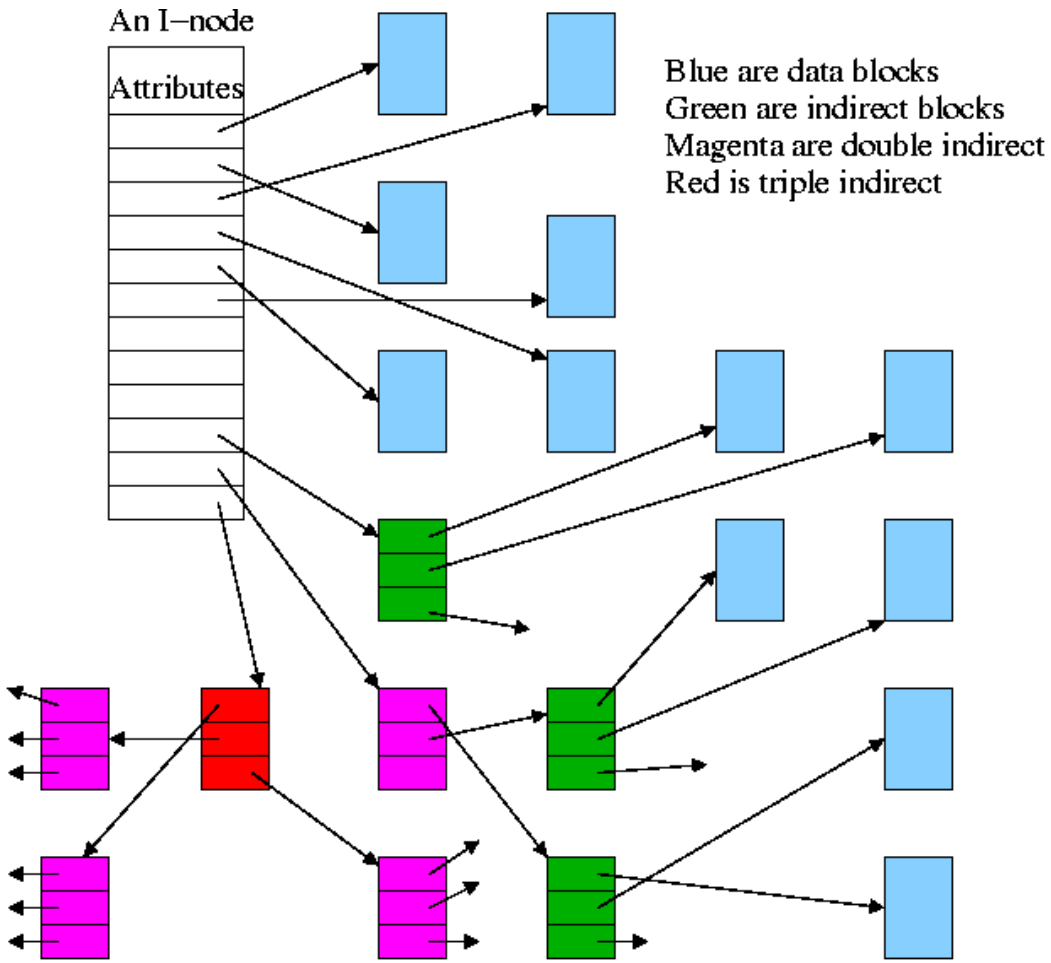
Nós índice Cada arquivo irá reservar um nó índice. No nó índice são armazenadas informações a respeito do arquivo.

Clusters de dados Segue a área de armazenamento de dados.

A estrutura típica do nó índice é a seguinte:

Campo	Descrição
Número	número do nó índice
Dono	número identificador do dono do arquivo
Grupo	número identificador do grupo do arquivo
Tipo	os tipos principais são arquivo e diretório
Permissões	A lista de permissões, armazenada no formato *rwx*
Última modificação	data/hora da última modificação
Criação	data/hora da criação do arquivo
Tamanho	tamanho do arquivo
Número de links	número de links para este arquivo
Lista de ponteiros	Tipicamente, 10 ponteiros diretos, 1 indireto duplo, 1 indireto triplo

Nós índice representando arquivos:



Como descrito, tudo no sistema Unix é armazenado em uma estrutura única de arquivo. Os diretórios são representados através de um arquivo que contém uma estrutura especial de diretório. A estrutura de diretório é bastante simples: cada registro possui dois campos, o número do nó índice e o nome do arquivo. Desta forma, o sistema pode representar arquivos de grande tamanho, com acesso rápido e possibilidade de implementação de segurança.
A estrutura de diretório utilizada pelo sistema de arquivos mais utilizado no Linux, *ext4*, é colocada a seguir:

```
struct ext4_dir_entry {
        unsigned long inode;     /* Inode number */
        unsigned short rec_len;  /* Directory entry length */
        unsigned short name_len; /* Name length */
        char name[up to 255];    /* File name */
};
```

Journaling filesystems

Um problema causado por uma falha na alimentação elétrica ou de hardware pode levar o sistema de arquivos a um estado inconsistente. A solução tradicional é utilizar um utilitário que recupera a integridade do sistema, como *fsck*, *icheck*, *dcheck*, etc..
No entanto, com uma quantidade grande de arquivos em um disco grande, o teste de consistência pode demorar um longo tempo; normalmente mais do que os usuários do sistema podem esperar.
Um Journaling filesystem possui um Jounal (ou *log*) que registra todas as transações *antes* delas ocorrerem verdadeiramente no sistema de arquivos. Após o eventual crash do sistema, pode-se identificar quais operações não foram efetivamente concluídas e retornar o sistema de arquivos a um estado consistente rapidamente.
No Linux, o *ext3* e *ext4* implementam o conceito. O NTFS também utiliza o conceito.

NTFS

Outro sistema de arquivos bastante importante é o disponibilizado pelos sistemas operacionais mais recentes da Microsoft: o NTFS (*NT File System*.
O NTFS apresenta uma série de vantagens sobre a FAT, onde pode-se citar os principais:

- Inclui capacidades de implementação de segurança que não existiam na FAT. Se a FAT for utilizada em um sistema com múltiplos usuários, não há como limitar o acesso a arquivos do sistema.
- NTFS é superior quando se está trabalhando com discos de grande capacidade (embora a FAT-32 tenha um desempenho bom);
- NTFS possibilita maior economia de espaço em disco.

O NTFS possibilita a criação de clusters de 512 a 64K bytes, embora o cluster de 4K bytes seja considerado um padrão.
O NTFS divide a partição em 3 partes: o registro de boot (já descrito), uma região chamada MFT (*Master File Table* e os clusters de dados (Figura 4).

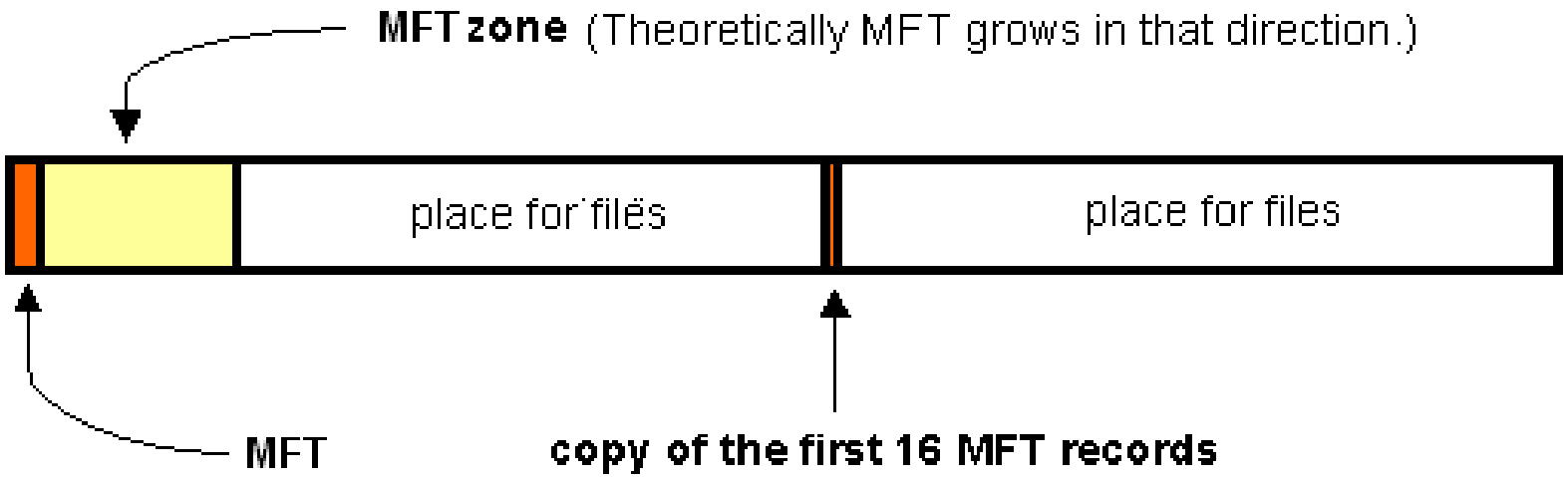


Figura : Regiões da partição do NTFS

A MTF ocupa cerca de 12% da área do disco e é sempre mantida livre, não importando se está ocupada ou não. Isso é feito para evitar-se a fragmentação da área mais importante do sistema de arquivos.
A MTF é separada em um conjunto de registros, cada registro representando um determinado arquivo. Cada registro está estruturado de acordo com a estrutura mostrada na Figura 1.

Standard info	File or dir name	Data or index	Unused

Tabela : Formato do registro da MFT

Os primeiros arquivos são arquivos de sistema, que tem como objetivo identificar os clusters livres, implementar o conceito de journal ou implementar quotas de uso de disco. Estes arquivos e seus significados são mostrados na Figura a seguir.

System File	File Name	Record	Purpose of the File
Master file table	$Mft	0	Contains one base file record for each file and folder on an NTFS volume. If the allocation information for a file or folder is too large to fit within a single record, other file records are allocated as well.
Master file table 2	$MftMirr	1	A duplicate image of the first four records of the MFT. This file guarantees access to the MFT in case of a single-sector failure.
Log file	$LogFile	2	Contains a list of transaction steps used for NTFS recoverability. Log file size depends on the volume size and can be as large as 4 MB. It is used by Windows NT/2000 to restore consistency to NTFS after a system failure.
Volume	$Volume	3	Contains information about the volume, such as the volume label and the volume version.

System File	File Name	Record	Purpose of the File
...	...	...	...
Attribute definitions	$AttrDef	4	A table of attribute names, numbers, and descriptions.
Root file name index	$	5	The root folder.
Cluster bitmap	$Bitmap	6	A representation of the volume showing which clusters are in use.
Bad cluster file	$BadClus	8	Contains bad clusters for the volume.
Security file	$Secure	9	Contains unique security descriptors for all files within a volume.
Upcase table	$Upcase	10	Converts lowercase characters to matching Unicode uppercase characters.
NTFS extension file	$Extend	11	Used for various optional extensions such as quotas, reparse point data, and object identifiers.
		12-15	Reserved for future use.

Quando um arquivo possui tamanho pequeno (tipicamente menor que 900 bytes), o conteúdo do arquivo será armazenado na própria MFT. Isso melhora o desempenho do sistema porque não é necessário gastar espaço extra no disco. Quando o tamanho do arquivo é grande, ao invés de armazenar o conteúdo do arquivo na própria MFT, o sistema utiliza a região como índices para os blocos do arquivo (de maneira semelhante aos nós índice do Unix). Desta forma, se o arquivo for muito grande e o espaço de índices do registro não for suficiente, uma série de ponteiros indiretos para clusters fora da MFT serão utilizados. O tamanho do arquivo só será limitado pelo tamanho do disco.

Permissões em um sistema NTFS

Permissão	Diretórios: um usuário pode:	Arquivos: um usuário pode:
Read (R)	Display folder: names, attributes, owner and permissions	Display file: data, attributes, owner, and permissions
Write (W)	Add files and folders, change a folder's attributes, and display owner and permissions	Change file attributes, create data in and append data to a file display owner and permissions
Execute (X)	Display folder attributes, make changes to folders within a folder, display owner and permissions	Display file attributes, owner and permissions, Run a file if it is an executable
Delete (D)	Delete a folder	Delete a file
Change Permission (P)	Change a folder's permissions	Change a file's permissions
Take Ownership (O)	Take ownership of a folder	Take ownership of a file

Comparação entre os principais sistemas de arquivos

Sistema de Arquivos	Criador	Ano de criação	Sistema operacional original
FAT12	Microsoft	1977	Microsoft Disk BASIC
FAT16	Microsoft	1988	MS-DOS 4.0
NWFS	Novell	1985	NetWare 286
HPFS	IBM & Microsoft	1988	OS/2
NTFS	Microsoft	1993	Windows NT
ext2	Rémy Card	1993	Linux
JFS	IBM	1990	AIX
FAT32	Microsoft	1996	Windows 95b
ext3	Stephen Tweedie	1999	Linux
Reiser4	Namesys	2004	Linux

Sistema de Arquivos	Tamanho máximo do nome do arquivo	Caracteres permitidos	Tamanho máximo do arquivo	Tamanho máximo do volume
FAT12	255 bytes	Any Unicode except NULL	32MB	1MB to 32MB
FAT16	255 bytes	Any Unicode except NULL	2GB	16MB to 2GB
FAT32	255 bytes	Any Unicode except NULL	4GB	512MB to 2TB
NTFS	255 char	Any Unicode except NULL	16TB	256TB
HPFS	255 bytes	Any byte except NULL	4GB	2TiB
ext2	255 bytes	Any byte except NULL	16GB to 2TB	2TiB to 32TiB
ext3	255 bytes	Any byte except NULL	16GB to 2TB	2TiB to 32TiB
JFS	255 bytes	Any byte except NULL	8EB	512TB to 4PB

Sistema de Arquivos	Permissões Posix	Links	Block Journaling	Meta-data Journaling	Case Sensitive	Transparent compression	Extents
FAT12	Não	Não	Não	Não	Não	Não	Não
FAT16	Não	Não	Não	Não	Não	Não	Não
FAT32	Não	Não	Não	Não	Não	Não	Não
NTFS	Não	Sim	Não	Sim	Sim	Sim	Sim
HPFS	Sim	Sim	Não	Não	Sim	Não	Sim
ext2	Sim	Sim	Não	Sim	Sim	Não	Não
ext3	Sim	Sim	Sim	Sim	Sim	Não	Não
JFS	Sim	Sim	Não	Sim	Sim	Não	Sim

Obtendo informações sobre o disco

```
# sudo smartctl -H /dev/sda
=== START OF INFORMATION SECTION ===
Model Family:     Seagate Barracuda 7200.12
Device Model:     ST31000524AS
Serial Number:    6VPK3RQM
LU WWN Device Id: 5 000c50 04ad7facf
Firmware Version: JC4A
User Capacity:    1,000,204,886,016 bytes [1.00 TB]
Sector Size:      512 bytes logical/physical
Rotation Rate:    7200 rpm
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   ATA8-ACS T13/1699-D revision 4
SATA Version is:  SATA 3.0, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is:    Mon Nov 16 20:23:44 2015 BRST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled

# sudo tune2fs -l /dev/sda5 | grep -i 'block size'
Block size:               4096
```

Partições

O programa *fdisk* pode ser utilizado para listar ou modificar a configuração das partições de cada disco. Exemplo:

```
#sudo fdisk /dev/sda

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders, total 1953525168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd272b344

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               63       80324       40131   de  Dell Utility
/dev/sda2   *       81920    27664383    13791232    7  HPFS/NTFS/exFAT
/dev/sda3        27664384   222976883    97656250    7  HPFS/NTFS/exFAT
/dev/sda4       222978046  1953523711   865272833    5  Extended
/dev/sda5       222978048  1199540223   488281088   83  Linux
/dev/sda6      1199542272  1883133951   341795840   83  Linux
/dev/sda7      1883136000  1953523711    35193856   82  Linux swap / Solaris
```

Montando dispositivos

- Para que dispositivos de armazenamento sejam acessados no sistema Unix, eles necessitam ser *montados* como parte do sistema de arquivos.
- Isso é feito através do comando *mount*.
- Para verificar as unidades montadas, simplesmente digite o comando *mount*.
- Para verificar as unidades e os pontos de montagem utilizadas na inicialização do sistema, veja o conteúdo do arquivo /etc/fstab.

Comando df

```
df [-aikT] [sistArq]
```

`i`	imprime a ocupação de inodes do disco
`k`	imprime a alocação total em kbytes
`T`	imprime o tipo de sistema de afrquivo
`h`	para ser lido por humanos...

Exemplo:

```
# df -T
Filesystem     Type     1K-blocks      Used Available Use% Mounted on
/dev/sda5      ext4     480487128 114172308 341884384  26% /
/dev/sda3      fuseblk   97656244  39489648  58166596  41% /windows
/dev/sda6      ext4     336301224 179867088 139327960  57% /home
/dev/sr0       iso9660      25500     25500         0 100% /media/pedroso/Cisco
```

A interpretação do resultado acima é muito importante. Anote as observações dadas em aula.

Montando dispositivos

Todos os dispositivos devem ser montados na árvore de diretórios do sistema para serem utilizados. Para montar dispositivos no sistema de arquivos utiliza-se o comando mount.

```
#sudo mount /dev/sda1 win
#df -T
Filesystem     Type     1K-blocks      Used Available Use% Mounted on
/dev/sda5      ext4     480487128 114172288 341884404  26% /
/dev/sda1      ntfs      97656244  39489648  58166596   41% /home/pedroso/win
/dev/sda6      ext4     336301224 179846428 139348620  57% /home
/dev/sr0       iso9660      25500     25500         0 100% /media/pedroso/Cisco
/dev/sda1      vfat         40034       116     39918   1% /home/pedroso/Dropbo
```

Características do sistema de arquivos do Unix

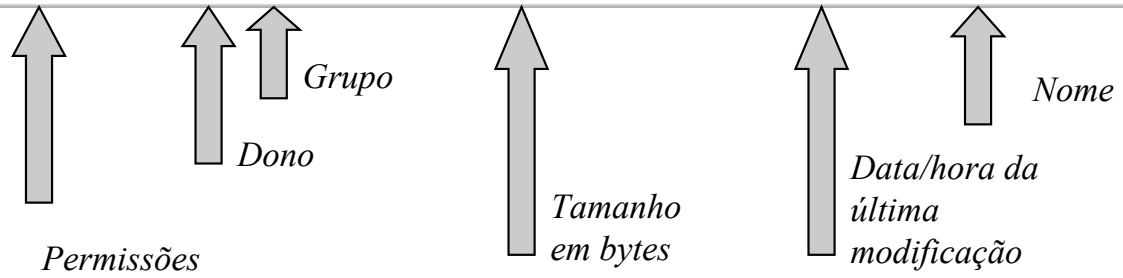
Estrutura de diretórios do sistema. Diretórios comuns utilizados:

/	Diretório raiz
/boot	Arquivos de boot
/bin	Comandos essenciais
/dev	Arquivos de dispositivos
/etc	Arquivos de configuração do sistema
/home	Arquivos dos usuários do sistema
/lib	Bibliotecas compartilhadas
/mnt	Diretório para montar partições temporariamente
/proc	Informações sobre processos do sistema
/root	Diretório home do administrador do sistema
/sbin	Arquivos executáveis essenciais ao sistema
/tmp	Arquivos temporários
/usr	Outra hierarquia secundária
/var	Dados variáveis

Permissões de acesso

```
nomehost:/etc$ ls -l
total 11
```

lrwxrwxrwx	root	root	9	Dec 9 14:01	rmt -> /sbin/rmt*
-rw-r--r--	root	root	743	Jul 31 1994	rpc
-rw-r--r--	root	root	86	Jan 28 1994	securette
-rw-r--r--	root	root	21394	Dec 9 14:22	sendmail.000
-rw-r--r--	root	root	23580	Jan 6 12:28	sendmail.cf
drwxr-xr-x	root	root	1024	Dec 9 13:59	skel/
-rw-r--r--	root	root	314	Jan 9 1995	slip.hosts
-rw-r--r--	root	root	342	Jan 9 1995	slip.login



Permissões de Acesso

Nas permissões de acesso do arquivo, *drwxrwxrwx* significa:

1. O primeiro caracter informa o tipo de arquivo (d-diretório,l-link, etc.)
2. Os três caracteres seguintes indicam as permissões do proprietário, no formato *rwx*: r-leitura, w-escrita e x-execução;
3. Os próximos três caracteres seguintes indicam as permissões do grupo, no formato *rwx*: r-leitura, w-escrita e x-execução;
4. Os últimos três caracteres indicam as permissões de todos, no formato *rwx*: r-leitura, w-escrita e x-execução;

A identificação do arquivo poderá ser:

- `-` Arquivo comum;
- `d` Diretório;
- `l` Link simbólico;
- `b` Arquivo especial de bloco;
- `c` Arquivo especial de caracter;
- `p` Arquivo especial de comunicação inter processo (“pipe”);
- `s` Arquivo especial que identifica um socket.

Tipos de arquivos

- Arquivos comuns são arquivos contendo dados;
- Diretórios são arquivos binários consistindo de uma lista de outros arquivos que ele contém;
- Arquivos de caracteres correspondem a acesso a dispositivos baseado em caracteres que não é buferizado);
- Os arquivos especiais de blocos correspondem a acesso a dispositivos com I/O de blocos Links são mecanismos que permitem vários nomes de arquivo referirem a um único arquivo no disco;
- Sockets são um tipo especial de arquivo utilizado para comunicação entre processos.

O diretório /dev

- Não contém arquivos reais.
- É uma interface para os device drivers do sistema;
- Permite que os processos se comuniquem com os device drivers utilizando chamadas padrão do sistema de arquivos, o que simplifica os procedimentos de E/S dos aplicativos;
- O diretório /dev é um sistema de arquivos especial.
- Usado por sistemas Unix, Windows e OS2.
- Sockets são um tipo especial de arquivo utilizado para comunicação entre processos.

O diretório /dev

Alguns nomes usados:

- lp line printers
- pt pseudo-terminals (virtual terminals)
- tty terminals
- sd SCSI driver, also used by libATA (modern PATA/SATA driver), USB, IEEE 1394, etc.
  - sda first registered device
  - sdb second registered device
- ttyS serial port driver
- ttyUSB USB serial
- sr SCSI CD-ROM device

O diretório /proc

- Não contém arquivos reais;
- Contém informações a respeito do estado do sistema: uso de memória, dispositivos montados, configuração de hardware, configuração de rede, etc.;
- É um centro de controle e informação do kernel.
- Os processos podem acessar informações do estado atual do kernel usando operações de leitura de arquivos.
- Sockets são um tipo especial de arquivo utilizado para comunicação entre processos.

O diretório /proc

Exemplo:

```
#cat /proc/meminfo
MemTotal:        8053576 kB
MemFree:          970012 kB
Buffers:          351788 kB
Cached:          2061740 kB
SwapCached:            0 kB
Active:          5077556 kB
Inactive:        1689132 kB
Active(anon):    4355952 kB
Inactive(anon):   769452 kB
Active(file):     721604 kB
Inactive(file):   919680 kB
Unevictable:          80 kB
Mlocked:              80 kB
SwapTotal:     35193852 kB
SwapFree:      35193852 kB
Dirty:               476 kB
Writeback:             0 kB
AnonPages:       4353364 kB
Mapped:           602408 kB
Shmem:            772252 kB
...
```

Comando cd

Troca de diretório. Sintaxe:

```
cd [diretório]
```

Caso o diretório seja omitido, o diretório corrente será o diretório raiz do usuário.

```
# cd
# cd /var/log
```

Exibe o diretório corrente. Exemplo:

```
# pwd /usr/bin
```

Comando ls

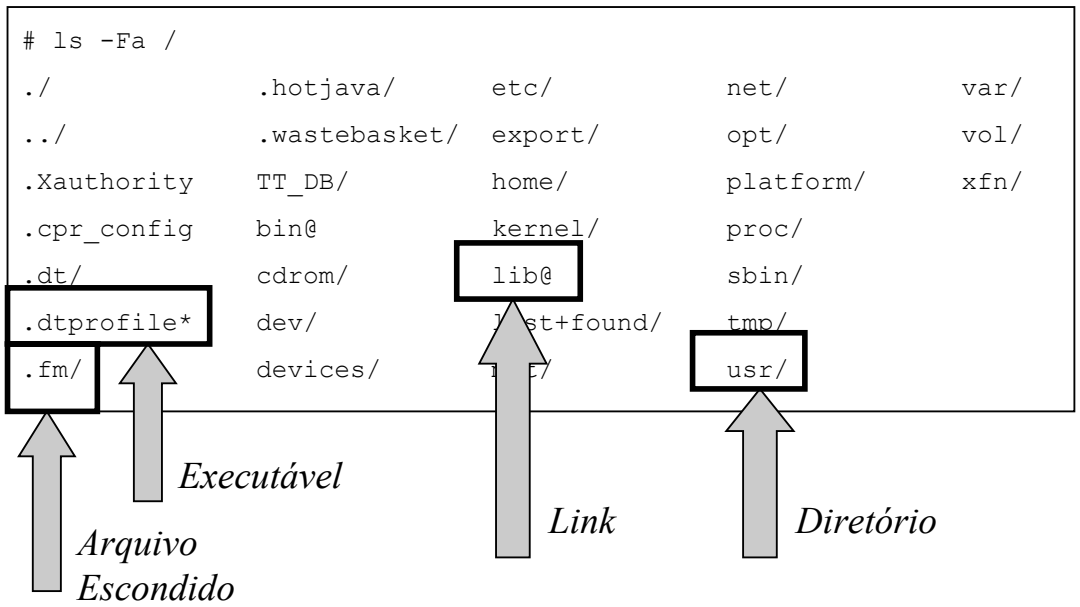
Lista conteúdo de diretório.

```
ls [-lastdg] [diretório]
```

`s`	formato longo, com permissões, data de criação, dono e grupo
`a`	mostra todos os arquivos, inclusive os escondidos (iniciam com ".")
`s`	informa o tamanho do arquivo em blocos do sistema de arquivos
`t`	ordena por ordem cronológica de criação
`d`	lista os diretórios
`g`	lista o grupo do arquivo
`L`	informa quais arquivos são executáveis, diretórios ou links através de marcas especiais: @ identifica um link, * identifica um executável, / identifica um diretório

Exemplo de uso do comando *ls*

As opções podem ser conjugadas em um único comando. Pode ser utilizado o caracter curinga (*) para selecionar os arquivos.



Exercício

1. Verifique as permissões do arquivo /etc/passwd;
2. Verifique as permissões de todos os arquivos abaixo da raiz (/);
3. Verifique as permissões no diretório /home;

Comando mkdir

Cria Diretório. Exemplo:

```
# cd /tmp
# mkdir teste
# pwd
/tmp
#
```

Comando who

Verifica quais são os usuários que estão com sessão ativa no sistema

```
who [am i]
```

Exemplo:

```
# who
# who am i
```

Comando more

Exibe o conteúdo de arquivos nomeados, fazendo pausas a cada tela cheia.

```
more [arquivo 1] [arquivo 2] ...  [arquivo n]
```

Exemplo:

```
# more /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
```

Comando tail

Exibe o final de um arquivo. Sintaxe:

```
tail [+|- n] filename
```

`-f`	exibe continuamente
`-n`	exibe as últimas n linhas
`...`	ver manual para mais opções

Exemplo:

```
# tail -20 /var/log/messages
# tail -f /var/log/messages
```

Comando cp

Copia arquivos para um outro arquivo ou diretório. Sintaxe:

```
cp <arqOrigem> <arqDestino>
```

ou

```
cp <arqOrigem1> <arqOrigem2> ...  <dirDestino>
```

Exemplo:

```
# cd /tmp
# mkdir dt
# cp /etc/hosts /etc/services dt
# cd dt
# ls
```

Comando mv

Move arquivos para um outro arquivo ou diretório. O(s) arquivo(s) origem deixa(m) de existir. Sintaxe:

```
mv <arqOrigem> <arqDestino>
```

ou

```
mv <arqOrigem1> <arqOrigem2> ...  <dirDestino>
```

Comando passwd

Modifica a senha pessoal. Sintaxe:

```
passwd [-fs] [username]
```

f	troca o nome completo associado ao usuário
s	troca o shell do usuário

Exemplo:

```
# passwd
passwd:  Changing password for cpedroso
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for cpedroso
```

Obs. *O superusuário (root) poderá alterar a senha de outros usuários*

Comando rm

Apagar arquivos. Sintaxe:

```
rm [rif] <arquivo 1> ...  <arquivo n>
```

- `r` Apaga todos os diretórios e arquivos recursivamente a partir do diretório especificado (USE COM CUIDADO!!!)
- `i` Solicita confirmação do usuário
- `f` Não solicita confirmação ao usuário

Exemplo:

```
# rm -i teste*
rm: teste is a directory
rm: remove teste1 (yes/no)? y
rm: remove teste2 (yes/no)? y
rm: remove teste3 (yes/no)? y
```

Comando chown

```
chown [ -fhR ] owner [ :  group ] file ...
```

- `f` se o arquivo é um link simbólico, troca o dono do link
- `h` não informa erros
- `R` recursivo

Exemplo:

```
# ls -ls hosts
   8 -r--r--r--   1 root     other          77 Oct 12 16:43 hosts
# chown cpedroso hosts
# ls -ls hosts
   8 -r--r--r--   1 cpedroso other          77 Oct 12 16:43 hosts
```

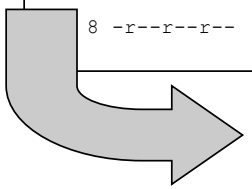
Comando chmod

Modifica as permissões de um arquivo ou diretório.

```
chmod [ -fR ] <absolute-mode> file ....
```

Modo absoluto: o usuário informa a nova permissão através de 3 números quando transformados para binário representam as permissões. Exemplo:

```
# ls -ls
total 16
   8 -r--r--r--   1 cpedroso other         77 Oct 12 16:43 hosts
   8 -r--r--r--   1 root     other       2947 Oct 12 16:43 services
# chmod 753 hosts
# ls -ls
total 16
   8 -rwxr-x-wx   1 cpedroso other         77 Oct 12 16:43 hosts
   8 -r--r--r--   1 root     other       2947 Oct 12 16:43 services
```



Nova permissão = 753 (dono – grupo – todos)

Dono: 7 = 111 (rwx)

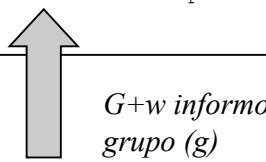
Grupo: 5 = 101 (r-x)

Todos: 3 = 011 (-wx)

Comando chmod

Pode ser utilizado o operador +/- para modificar permissões.
Exemplo:

```
# ls -ls
total 8
   8 -rwxr-x-wx   1 cpedroso other         77 Oct 12 16:43 hosts
# chmod g+w hosts
# ls -ls
total 8
   8 -rwxrwx-wx   1 cpedroso other         77 Oct 12 16:43 hosts
```



G+w informou que a permissão de escrita (w) foi concedida ao grupo (g)

u(usuário corrente); g (grupo); o (todos)

r(leitura); w(escrita); x(execução)

Comando grep

Exibe todas as linhas, dos arquivos especificados, que contém um certo padrão.

```
grep [-ilvc] +padrão <arquivo1> <arquivo2> ...
                    <arquivoN>
```

- `i` ignora upercase/lowercase
- `l` imprime somente os nomes dos arquivos contendo o padrão
- `c` imprime somente a contagem de ocorrências do padrão
- `v` inverte a busca

Comando grep

Exemplo. O comando a seguir procura a palavra "host" em todos os arquivos que iniciam com a string "ns" do diretório /etc:

```
# grep host /etc/ns*
nscd.conf:#     Currently supported cache names: passwd, group, hosts
nscd.conf:#     enable-cache            hosts           no
nscd.conf:      positive-time-to-live   hosts           3600
nscd.conf:      negative-time-to-live   hosts           5
nscd.conf:      suggested-size          hosts           211
nscd.conf:      keep-hot-count          hosts           20
nscd.conf:      old-data-ok             hosts           no
nscd.conf:      check-files             hosts           yes
nsswitch.conf:# "hosts:" and "services:" in this file are used only if the
```

Comando ln

Cria links simbólicos. Sintaxe:

```
ln [ -fns ] sourceFile [target]
```

- `f` Não pergunta ao usuário
- `s` Cria um link simbólico. Caso esta opção não seja utilizada, será criado um hard link
- `n` se o alvo é um arquivo existente, não apaga seu conteúdo

link simbólico- continuação

- Soft Link: é um inode que aponta para o arquivo alvo, criando um novo nome para o link. Se o soft link for apagado, o arquivo original é preservado.
- Hard Link: é uma entrada no diretório que aponta para o mesmo inode do arquivo. Se o hard link for apagado, o arquivo original é preservado.

link simbólico- continuação

Exemplo de criação de soft link:

```
# pwd
/tmp/dt
# ln -sn /etc/passwd teste
# ls -ls
total 24
   8 -r-xrwx--x   1 cpedroso other         77 Oct 12 16:43 hosts
   8 -r--r--r--   1 root     other       2947 Oct 12 16:43 services
   8 lrwxrwxrwx   1 root     other         11 Oct 12 17:25 teste -> /etc/passwd
```

No exemplo: o arquivo teste é um link simbólico para o arquivo /etc/passwd.

link simbólico- continuação

- Exemplo de criação de hard link:

```
# pwd
/tmp/dt
# echo 'Teste 123' > teste.txt
# ln teste.txt outroarq.txt
# ls -li *txt

26738767 -rw-r--r-- 2 root root 15 Oct  1 15:30 outroarq.txt
26738767 -rw-r--r-- 2 root root 15 Oct  1 15:30 teste.txt
```

- Note o número do inode, 26738767, igual para os dois arquivos.

```
# rm teste.txt
# ls -lsi outroarq.txt

26738767 -rw-r--r-- 2 root root 15 Oct  1 15:30 outroarq.txt
```

comando du

- Exibe o espaço ocupado de um diretório e de todos os seus subdiretórios. Exemplo:

```
# du -h --max-depth=1
3.1M ./Slides
3.7M ./Trabalho
2.1M ./Exercicios
```

- A opção "–max-depth" indica a profundidade na área de diretórios a ser impressa. No resultado do comando: 3.1 M, 3.7 M e 2.1 M representam o espaço em disco, em bytes. Outros formatos de saída podem ser escolhidos. Consulte a página de manual para as outras opções.

Pipes

- Pipe é uma valiosíssima característica do sistema UNIX que permite que vários programas sejam concatenados. Deste modo, o output de um programa é input do próximo. Esse recurso é muito utilizado para aplicar filtros.
- Por exemplo, para listar os processos de um determinado usuário, manda-se listar os processos e a saída deste comando serve de entrada para um comando que filtre nesta lista todas as ocorrências do usuário procurado. Exemplo:

```
# ls -ls /etc |grep ns
# cat /etc/nsswitch.conf|grep host
# cat /etc/nsswitch.conf|more
```

Redirecionamento

- A saída produzida por um programa será escrita em um arquivo ao invés de ser impressa no vídeo.
- Exemplo 1. Será criado a arquivo teste, com o conteúdo do arquivo /etc/hosts:

```
#cd /tmp
#cat /etc/hosts > teste
```

- Exemplo 2. O arquivo /etc/passwd será concatenado com o arquivo teste:

```
#cat /etc/passwd  >> teste
```